FA24: APPLIED ALGORITHMS: 11565 - ASSIGNMENT - 1

QUESTION 2

a)
```
def function1 (n):
    for i in range (0,n):            ——— ①
        for j in range (0, i):       ——— ②
            K = j
            while K < n:             ——— ③
                K += 1
    return
```

① The outer loop runs $\boxed{n \text{ times}}$

② In the middle loop, for each value of i, the loop runs i times

Total iterations $\Rightarrow \sum_{i=0}^{n-1} i = 0 + 1 + 2 + \dots (n-1)$

$$= \frac{n(n-1)}{2} \Rightarrow \boxed{O(n^2)}$$

i	mid loop
0	0
1	1
2	2
i	n-1

③ In the inner while loop. for each iteration of the middle loop. the while loop runs from $K = j$ to $K = n$ times $\Rightarrow n - j$ times

Total iterations $\Rightarrow \sum_{i=0}^{n-1} \sum_{j=0}^{i-1} (n-j) = \sum_{i=0}^{n-1} \left[n \times i - \sum_{j=0}^{i-1} j \right]$ $\to \frac{i(i-1)}{2}$

$$= \sum_{i=0}^{n-1} \left[n.i - \cancel{\frac{i(i-1)}{2}} \right] = \sum_{i=0}^{n-1} n.i$$

$$= n \times \frac{n(n-1)}{2} = \boxed{O(n^3)}$$

Hence the overall time complexity will be $O(n^3)$

b) def function2(n):

```
i = n
while i > 0:
    i = i // 3
return
```

$\Rightarrow$ Variable i starts at $i = n$ and is divided by 3 in each iteration and the loop continues until $i > 0$

Total number of iterations:

i	iteration
1	$n/3$
2	$n/3^2$
⋮	
K	$n/3^k$

- In each iteration i is divided by 3 $\Rightarrow \frac{n}{3}$
- After K iterations, i will be $\Rightarrow \frac{n}{3^k}$
- Loop terminates when $\frac{n}{3^k} \leq 1$

$$\Rightarrow K \approx \log_3 n$$

$\frac{n}{3^k} = 1$

$n = 3^k$

$\log_3 n = k$

- Hence the total number of iterations is proportional to $\log_3 n$

$\boxed{\text{Thus the overall time complexity will be } O(\log n)}$

c) def function3(n):

```
i = 1
while i * i < n:
    i += 1
return
```

The while loop continues as long as $i^2 < n$, which implies $i \times i$ becomes greater than or equal to n.

- The loop increments i by 1 in each iteration
- Condition $i^2 < n \Rightarrow$ ~~i <~~ $i \geq \sqrt{n}$
- Hence the loop runs approximately $\sqrt{n}$ times.

$\boxed{\text{Thus the overall time complexity will be } O(\sqrt{n})}$

d) def function4(n):

```
i = n
while i > 0:          ——①
    for j in range (0, n):     ——②
        for k in range (0, j):     ——③
            x = k
    i = i // 2
return
```

① The outer while loop starts with $i = n$ and repeatedly halves i until it reaches 0.

Hence it runs $O(\log n)$ times

② The middle loop (for) runs n times for each while loop iteration

③ For each value of j, the inner for loop runs j times

j	no. of times
0	0
1	1
2	2
j	n-1

Total numbers of iterations for the nested for loops:

$$\sum_{j=0}^{n-1} j = 0+1+2+\cdots+(n-1) = \frac{n(n-1)}{2} \Rightarrow O(n^2)$$

$$\boxed{\text{Total time complexity: } O(\log n) \times O(n^2) = O(n^2 \log n)}$$

e) def functions5(n):

```
i = 2
while i < n:
    i = i * i
return
```

The variable i starts at 2 and is squared in each iteration.

The while loop continues as long as $i < n$

- In each iteration i is squared,
  So the value of i grow as:

  First iteration $i = 2^2 = 4$

  Second iteration $i = 4^2 = 16$

  Third iteration $i = 16^2 = 256$ and so on

i	i^2	k iters
2	2^2	2^{2^1}
4	4^2	2^{2^2}
16	16^2	2^{2^3}
		⋮

k^{th} iteration $\Rightarrow 2^{2^k}$

- So after k iterations i will be 2^{2^k}
- Solving for k $\Rightarrow$ $2^{2^k} \geq n$

  Taking log twice...

  $k \approx \log \log n$

$$\boxed{\text{Hence the overall time complexity will be } O(\log \log n)}$$

SUBMITTED BY
SUBASHREE V S
subavenk@iu.edu
2001429203

QUESTION-11

a) $f(n) \in O(g(n))$ where $f(n) = 3n + 4$ and $g(n) = n$

Big-O Definition:
we say that $f(n) \in O(g(n))$ if there exist positive constants c and n_0 such that for all $n \geq n_0$: $f(n) \leq c.g(n)$

PROOF:

$f(n) = 3n + 4$

$g(n) = n$

we want to find constants c and n_0 such that: $3n + 4 \leq c.n$ for all $n \geq n_0$

Dividing both sides by n, assuming $n > 0$: $3 + \frac{4}{n} \leq c$

As $n \to \infty$, $\frac{4}{n}$ becomes very small, so for sufficiently large n, we can make the inequality hold.

Let $n_0 = 1$, then $3 + \frac{4}{1} = 7$

Thus if we choose $c = 7$, the inequality $3n + 4 \leq 7n$ holds for all $n \geq 1$

Hence we conclude that $f(n) = 3n + 4 \in O(n)$

$\boxed{\therefore f(n) \in O(g(n)) \text{ is true}}$

b) $f(n) \in \Theta(g(n))$ where $f(n) = 5n^2 + 2n \log n$ and $g(n) = n^2$

Big-Theta Definition:
We say $f(n) \in \Theta(g(n))$, if there exist positive constants c_1, c_2 and n_0 such that for all $n \geq n_0$: $c_1.g(n) \leq f(n) \leq c_2.g(n)$

PROOF:

$f(n) = 5n^2 + 2n\log n$

$g(n) = n^2$

we need to show that $f(n)$ is asymptotically bounded both above & below by $g(n)$

1. UPPER BOUND:

For large n, the term $5n^2$ dominates $2n\log n$ because of n^2 grows faster than $n\log n$

$\Rightarrow f(n) = 5n^2 + 2n\log n \leq 6n^2$ for sufficiently large n

So $f(n) \leq c_2 n^2$ with $c_2 = 6$

2. LOWER BOUND:

Similarly for large n, the term $5n^2$ dominates $2n\log n$

So, $f(n) = 5n^2 + 2n\log n \geq 5n^2$ for sufficiently large n

$\Rightarrow f(n) \geq c_1 n^2$ with $c_1 = 5$

Since we have both upper and lower bounds, we can conclude that $f(n)$ is asymptotically tight around n^2

$\Rightarrow f(n) = 5n^2 + 2n\log n \in \Theta(n^2)$

$\boxed{\therefore\ f(n) \in \Theta(g(n)) \text{ is true}}$

c) $f(n) \in \Omega(g(n))$ where $f(n) = n\log n + 100$ and $g(n) = n$

Big-Omega Definition:

we say $f(n) \in \Omega(g(n))$ if there exists positive constants c and n_0 such that for all $n \geq n_0$: $f(n) \geq c \cdot g(n)$

PROOF:

$f(n) = n\log n + 100$

$g(n) = n$

We want to find constants c and n_0 such that:

$n\log n + 100 \geq c.n$ for all $n \geq n_0$

For large n, nlogn dominates 100, so that 100 becomes negligible.

So focusing on term $n\log n$, we need $n\log n \geq c.n$

Divide on both sides by n, $\log n \geq c$

For large enough n, this inequality holds for any constant c.

Specifically for $n \geq 2$, we can choose $c = 1$ and $\log n \geq 1$ will be true

Since $f(n) \geq c.n$ for $c = 1$ and $n_0 = 2$, we can conclude that

$$f(n) = n\log + 100 \in \Omega(n)$$

$\boxed{\therefore\ f(n) \in \Omega(g(n)) \text{ is true}}$

d) $f(n) \in \Theta(g(n))$ where $f(n) = n\log n$ and $g(n) = 2^n$

PROOF:

$f(n) = n\log n$

$g(n) = 2^n$

For $f(n) \in \Theta(g(n))$, we would need $f(n)$ to be asymptotically both upper and lower bounded by $g(n)$. However $n\log n$ grows much slower than 2^n.

* UPPER BOUND

For ~~log~~ large n, $n\log n$ grows slower than 2^n. w.k.t $n\log n = o(2^n)$

because 2^n grows exponentially while $n\log n$ grows Sub exponentially

This implies that $f(n)$ is not comparable to $g(n)$ in terms of growth rates.

Since $n\log n$ grows much slower than 2^n, we have

$$f(n) = n\log n \notin \Theta(2^n)$$

$\boxed{\therefore\ f(n) \in \Theta(g(n)) \text{ is not true}}$

e) $f(n) \in o(g(n))$ where $f(n) = n^{1.5} + \log_2 n$ and $g(n) = n^2$

Little-o Definition:

We say $f(n) \in o(g(n))$ if for any constant $c > 0$, there exists n_0 such that for all $n \geq n_0$: $f(n) < c.g(n)$

PROOF:

$f(n) = n^{1.5} + \log_2 n$

$g(n) = n^2$

We need to show that $f(n)$ grows strictly slower than $g(n)$

For large n, $n^{1.5}$ dominates $\log_2 n$, so we can focus on comparing $n^{1.5}$ with n^2

$$\frac{f(n)}{g(n)} = \frac{n^{1.5} + \log_2 n}{n^2} = \frac{n^{1.5}}{n^2} + \frac{\log_2 n}{n^2} = \frac{1}{n^{0.5}} + \frac{\log_2 n}{n^2}$$

As $n \to \infty$

$$\frac{1}{n^{0.5}} \to 0 \quad \text{and} \quad \frac{\log_2 n}{n^2} \to 0$$

Thus $\frac{f(n)}{g(n)} \to 0$ as $n \to \infty$ which means that $f(n)$ grows strictly slower than $g(n)$

$\boxed{\therefore\ f(n) \in o(g(n)) \text{ is true}}$

SUBMITTED BY
SUBASHREE VS
Subavenk@iu.edu
2001429203